Title	Model Checking for Licensing Support in the Finnish Nuclear Industry
Author(s)	Pakonen, Antti; Valkonen, Janne; Matinaho, S; Hartikainen, M
Citation	International Symposium on Future I&C for Nuclear Power Plants (ISOFIC 2014), Jeju Island, Republic of Korea, 24 - 28 August 2014
Date	2014
Rights	This article may be downloaded for personal use only

Model Checking for Licensing Support in the Finnish Nuclear Industry

Antti PAKONEN[1], Janne VALKONEN[1], Sami MATINAHO[2], Markus HARTIKAINEN[2]

1. VTT Technical Research Centre of Finland, P.O. Box 1000, FI-02044 VTT, Finland (antti.pakonen@vtt.fi, janne.valkonen@vtt.fi)

2. Fortum Power and Heat, P.O. Box 1, FI-00048 FORTUM, Finland (sami.matinaho@fortum.com, markus.hartikainen@fortum.com)

Abstract: This paper examines how model checking can be used to support the qualification of digital I&C software in nuclear power plants, in a way that is consistent with regulatory demands – specifically, the common position of seven European nuclear regulators and authorised technical support organisations. As a practical example, we discuss the third-party review service provided by VTT for the power company Fortum in the I&C renewal project of the Loviisa plant in southern Finland.

Keywords: Model checking, verification and validation, digital I&C

1 Introduction

Reliability of digital instrumentation and control (I&C) software is a critical issue for nuclear power plants, both in new-builds and modernisation projects. The inherent complexity of software-based control systems means that the industry and regulatory bodies face challenges in ensuring that systems are error-free, and meet their requirements.

The Finnish Regulatory Guides on nuclear safety (YVL) state that with systems of considerable safety significance, "a safety assessment shall be carried out by an independent third-party organisation" [1]. Also, a common position of several European nuclear regulators – including the Finnish regulator STUK – is that for a software-based safety system "an independent assessment of the system is essential to provide the degree of confidence in the design process, in the product and in the personnel involved. The independent assessment ordered by the supplier of the system or by the licensee can be regarded as an important part of the evidence in the safety demonstration" [2].

The common position report also presents the consensus on the proper use of formal methods in licensing of safety critical software. While certainly useful and necessary, the use of formal methods is always challenging, and each method is only limited to a specific aspect. Inappropriate use of formal methods can even be dangerous, as lack of legibility may lead to difficulties in verification, and the impossibility of expressing all types of requirements may lead to incompleteness or inconsistencies [2].

To meet these demands and challenges, VTT has been providing an independent, third-party I&C software verification service based on model checking. Model checking is a formal, computer-assisted verification method that is used to exhaustively check that a model of a hardware or software system fulfils a set of formalised functional properties [3]. Due to the exhaustive analysis (taking into account all relevant model states or behaviours), model checking can reveal design faults that are difficult to find using more traditional verification and validation (V&V) methods.

In this paper, we introduce the rationale and the challenges in using model checking for the verification of I&C application software in nuclear power plants. We then discuss how model checking can support control software licensing in a way that is consistent with regulatory demands, using the common position report as a reference. The I&C renewal project of the Finnish nuclear power plant at Loviisa is used as a case example, as we introduce the work process used by VTT in the independent model checking service provided for the licensee Fortum.

2 Model checking of I&C software

2.1 Basics of model checking

Model checking [3] is a formal method that can be used to exhaustively prove that a model of a (software of hardware) system fulfils a specified property. A software tool called a model checker is used to verify that no state or execution of the system model violates any stated property. Typically, formalised properties can be divided in two main types [4]: *safety properties* state that an undesired scenario never happens, while *liveness properties* state that a desired scenario keeps happening.

If a model execution contrary to a property is found, it is indicated to the user as a counterexample (error trace). Analysis of the counterexample can then reveal a design error in the system, but also an error in the model or the property formulation. It can therefore be said that the method is self-repairing to a certain degree.

Despite the obvious advantage of exhaustive analysis, a key challenge in model checking is the state explosion problem. The number of possible model states grows exponentially as new model inputs and components are introduced [3]. The problem is addressed using symbolic verification, based on the manipulation of Boolean formulas. Several tools use binary decision diagrams (BDD) to allow the verification of systems whose extremely large state spaces would make it impossible to perform explicit enumeration [5]. Typically, the model is based on a finite state machine (FSM), and a temporal logic language is used for formulating the system properties.

The popular open source tool NuSMV [6] is a BDD-based symbolic model checker for verifying finite state systems, using a discrete representation of time. Properties can be specified using Linear Temporal Logic (LTL) or Computation Tree Logic (CTL) [3], and there is also limited support for the more human readable Property Specification Language (PSL) [7].

2.2 Rationale

Function blocks are a common programming language for implementing safety-classified I&C systems. Function block diagrams specify a clear input-output mapping, making it relatively easy to understand control flow. While the design of an individual elementary function block type can be proven error-free through rigorous unit testing, verification of complex function block diagrams is often only based on manual inspection and review. Testing and simulation can also be used, but the amount of input combinations, feedback loops, and internal memory makes it impossible to analyse all execution paths. Arguments for error-free software are then supported by referring to operational experience and quality of software development practices. Model checking, on the other hand, enables exhaustive (but still quite fast) analysis.

Model checking of I&C software function blocks has been an active research topic for at least 15 years [8][9]. In the nuclear industry, researchers of the Korea Advanced Institute of Science and Technology (KAIST) and the Konkuk and Korea universities in Seoul have applied model checking in the verification of function block based control software of the reactor protection system (RPS) of the APR-1400 reactor [10]. In Finland, VTT has been studying the use of model checking in the nuclear domain together with the Aalto University under the Finnish Research Programme on Nuclear Plant Safety since the year 2007 [11][12][13]. After early success in industrial pilot cases, the method was quickly put to practical use. VTT has consulted the Finnish Radiation and Nuclear Safety Authority (STUK) on evaluating NPP I&C system designs using model checking as early as 2008. When the Finnish Ministry of the Employment and the Economy launched the new nuclear safety research programme in 2010 and published the new framework plan, the use of formal modelling methods in nuclear power plant automation was stated to be one of the clear success stories in the previous research programme [14].

On safety-critical domains other than nuclear, model checking has also proven very useful. For a list of references, see, e.g., [11].

2.3 Challenges

Many practical challenges in model checking of function block diagrams have to do with the lack of dedicated tools for the domain. Research attempts on automating the process have focused on standard function block languages like the IEC 61131-3 [9], which are hardly universally adopted, especially in the

nuclear domain. Error-prone manual work and ad hoc solutions are therefore needed in construction of the model. A significant amount of work is also needed for the interpretation of counterexamples [15], which are visualised, at best, using trend graphs. VTT has solved these issues with in-house tool development based on NuSMV [9] (See chapter 5.4).

Formalising the properties to be verified is also a challenging issue. I&C requirements based on natural language can often be vague and ambiguous, whereas model checking depends on exact formal representation. Also, high expertise is needed to grasp the details of temporal logic. One proposed solution is to capture oft-occurring requirement constructs in templates or patterns that map a natural language representation to corresponding formulas suited for verification [16].

Thankfully, errors made in both system modelling and property formalisation are typically revealed through "false" counterexamples that demonstrate, e.g., unrealistic model behaviour, or a scenario that should not be contrary to a stated property, were the property correctly formulated.

2.4 Scope

When applying model checking to the verification of a function block diagram (based on non-standard, I&C system vendor-specific function blocks), it is important to note that what is being verified is that the way the blocks are used in the diagram does not result in unwanted execution paths. Other aspects of the design are outside the scope.

It is assumed that the elementary function blocks are free of errors, which can usually be proven via rigorous unit testing. Since the source code for the vendor-specific function blocks is typically not available (black box), the modelling is based on functional descriptions [9]. Even if the source code is available, it is likely that the programming language in use would make model checking challenging.

The code that is actually compiled and run on the hardware platform is based on lower-level languages such as C. It is assumed that the code generation will operate flawlessly, since any faults introduced at this point cannot be considered in model checking.

Fault tolerance is an important aspect of I&C software design in NPPs. While theoretically, the model can be constructed in the way that different failure mechanisms of the underlying I&C hardware are taken into account, such models easily become too complex. While specific issues can be checked by introducing additional model variables, systemic examination of hardware failure modes is currently unfeasible.

2.5 General limitations

Although model checking can be used to exhaustively prove that a model of a (software) design fulfils a given property, evidence gained with model checking is not a conclusive proof of a fault-free design. The model is representative of the actual system only to a certain degree (see chapter 2.6 for I&C specific issues).

Despite the extensive analysis, it is also difficult to guarantee that all necessary properties have been taken into account. The requirement specification serving as a starting point might not be complete, and effort is needed from the modeller to consider all necessary aspects.

It is also theoretically possible, although unlikely, that *both* the model *and* the properties contain errors that will hide an actual design error and result in a false positive.

Furthermore, although model checkers such as NuSMV are based on standard and well-known algorithms, the correctness of the model checker cannot be exhaustively proved.

2.6 I&C software specific limitations

Since the model is typically expressed as a type of finite state machine, model checking can only be effectively applied to systems that can easily be expressed in simple, discrete terms. For I&C, this means that designs containing arithmetically complex control loops (e.g. PID or model predictive control) cannot be verified. When it comes to more straightforward binary logic, however, the method scales very well. Analysis times for FSMs with as many as 10^{40} states are still in the range of minutes, if not seconds. Simple math is also not a problem for tools such as NuSMV. Problems can arise from, e.g., excessive use of feedback, or function blocks storing number data into memory, leading to state explosion.

When modelling function block diagrams, some aspects may require specific attention. For timing

blocks, the modeller needs to assign a certain number of model "cycles" to represent the length of the delay. Large values can easily lead to state explosion [10], while small values might prevent relevant execution paths.

Analogue signal values must also be discretised, since NuSMV can only handle integer data. For analogue inputs, the modeller should limit the possible values to avoid state explosion while, again, allowing for all relevant model executions. Some scaling back and forth of signal values is also sometimes necessary to properly model simple arithmetic.

(There exists model checkers other than NuSMV that support, e.g., real variables or continuous time, but VTT's experience has shown that such tools may not be as efficient for the verification of function block diagrams [13].)

It is noteworthy that it may be tempting to construct a closed loop model. By modelling also aspects of the environment (dynamics of the controlled plant), the state space can be reduced. Without an environment model, the analyst may also have to spend extra time processing counterexamples that are not possible in the context of the controlled process. The downside is that inaccuracy in the environment model may result in actual design errors ending up undetected. From a safety point of view, open loop analysis is more reliable.

3 Regulatory demands on the use of formal methods

At the request of the Western European Nuclear Regulators' Association (WENRA), a task force consisting of nuclear regulators and authorised technical support organisations of Finland, Sweden, Germany, UK, Spain and Belgium has identified common technical positions on licensing issues regarding the use of computer based systems in the implementation of safety functions in NPPs. Their report [2] is intended to guide the regulators' national policies on technical viewpoints, among other uses.

The U.S. Nuclear Regulatory Commission (NRC) has also participated in the meetings of the task force. Although the NRC does not officially endorse the report, it plans to continue informing the task force of NRC's technical positions, and informing the NRC of the task force's position on issues of mutual interest [2].

In the following, we individually address the common positions on the use of formal methods, as stated in section 1.9 of the report.

3.1 Common position 1.9.3.1

"No credit can be taken in a safety demonstration for the use "per se" of a formal method without due consideration being given to the specific evidence brought in by this use, and to its contribution to the safety demonstration of the system."

A design error revealed though model checking is a non-disputable piece of evidence. Concluding the value of a verification task that does not reveal any errors is less simple. Careful analysis of the inherent limitations is recommended (see chapters 2.5 and 2.6).

3.2 Common position 1.9.3.2

"Whatever (combination of) method(s) and notation(s) is used to describe the system requirements, this description shall be based on a definition of the system boundaries and on a systematic capturing of the functional and non-functional properties of the system. These boundaries and properties shall be explicitly, unambiguously and accurately documented."

The basic idea of model checking makes it absolutely necessary to explicitly and accurately define both the system requirements and the model boundaries.

As all verified requirements must be stated in terms of system model inputs, outputs, and internal variables, it may be difficult to systematically capture non-functional variables that cannot be accurately specified in those terms.

3.3 Common position 1.9.3.3

"Selection of methods and tools with respect to their intended application for formal descriptions and mathematical analysis shall be justified. The justification shall be in accordance with the safety demonstration."

The use of model checking is justified by the benefits over more conventional verification methods, as well as the applicability of the method for analysing function block diagrams (consisting mostly of binary logic). See chapter 2.2.

3.4 Common position 1.9.3.4

"There shall be objective evidence of a successful use of the formalisms and methods used in an application with comparable properties."

Model checking has been used in the verification of NPP I&C software in the Republic of Korea [10]. The method has also been successfully used for software verification in other safety-critical domains such as aviation or space. See chapter 2.2.

3.5 Common position 1.9.3.5

"The procedures and constraints for using the formal methods and tools shall be documented."

See chapters 5.3, 2.5, and 2.6.

3.6 Common position 1.9.3.6

"Any limitations of the formal descriptions, methods and tools used, and resulting descriptions, shall be explicitly documented. For example, any limitation in describing and reasoning about non-functional requirements, use of resources, or time critical events shall be stated."

See chapters 2.5 for limitations in general and chapter 2.6 for limitations in the context of NPP I&C software.

As all verified requirements must be stated in terms of system model inputs, outputs, and internal variables, it may be difficult to systematically capture non-functional variables that cannot be accurately specified in those terms.

3.7 Common position 1.9.3.7

"The formalisms and the methods used for specifying the system requirements shall be unambiguous and understandable by all technical staff involved."

In the verification service that VTT provides, the formalisation of the requirements into temporal logic is a task performed by VTT. Nevertheless, VTT experts have been educating Fortum staff on formalisms such as LTL and CTL.

3.8 Common position 1.9.3.8

"The formal description of the system requirements shall be validated against the results of a prior plant safety analysis, and of other relevant analyses at the plant level."

Due to the strict, specific formalism of temporal logic languages, it is difficult to find prior analysis that would support validation. Nevertheless, most semantic errors in requirement formalisation will be revealed through "false" counterexamples. Such a counterexample can, for example, demonstrate a scenario that would *not* be contrary to a *correctly* formalised property.

4 LARA: Loviisa NPP automation renewal project

Situated on the southern coast of Finland, and operated by the power company Fortum, the Loviisa NPP includes two pressurised water reactors of the type VVER-440, with a joint capacity of 976 MW. Loviisa 1 started operation in 1977 and Loviisa 2 in 1980. In 2012, Loviisa produced 7.61 TWh of electricity. Key figures measuring plant safety and performance reliability have been good throughout the operational history.

The old I&C systems at Loviisa consist of Russian and German technology from the 1970s. While the ageing of the systems had not yet produced any significant problems, concerns over the availability of spare parts and maintenance led to a decision in the early 2000s to start preparing the automation renewal project (LARA).

The new systems were to be delivered by a consortium of AREVA and Siemens, with safety I&C based on AREVA platforms such as TELEPERM XS, and operational I&C based on platforms of Siemens.

However, in May 2014, Fortum decided to discontinue the LARA project due to delays in the project implementation. At the same time, Fortum signed an agreement with Rolls-Royce regarding modernisation of the Loviisa Plant. Rolls-Royce will deliver all the required I&C systems, with the Finnish company Metso acting as a sub-supplier of non-safety systems. The aim is to implement the project by the end of 2018.

5 Use of model checking in LARA

5.1 Task description

Model checking was used to verify the correct functionality of application I&C software in LARA subsystems. Verification was performed by VTT as an

independent, third-party verification service commissioned by Fortum. Analysis was targeted at the selected I&C functions of each reviewed system.

The verification task was parallel with module level system tests carried out by the vendor.

Independently of the vendor, Fortum decided the I&C functions to be verified. VTT experts provided feedback and suggestions concerning which functions should be included. Fortum then supplied the necessary material (design sheets, requirement specifications, and other relevant documentation) to VTT.

VTT performed model checking, and reported any findings to Fortum. Fortum then analysed the findings – in particular assessing the safety relevance – and informed the vendor if necessary.

Finally, VTT was to prepare detailed verification reports that Fortum could have then submitted to the Finnish regulator (STUK) as part of LARA subsystem licensing. However, Fortum decided to discontinue the LARA project, and start a new modernisation project with Rolls-Royce. Fortum plans to continue using model checking in the new project.

5.2 Scope of verification

Verification has covered I&C functions of several LARA subsystems, including reactor protection, preventive protection, and emergency diesel generator functions. The analysed systems belong to the Finnish safety classes SC2 (the highest safety class where digital I&C can be applied) and SC3.

5.3 Work process

Based on previous experience on model checking NPP software, VTT has identified a set of practices and methods to efficiently and reliably carry out the verification process. In Fig. 1, the work process is described in a series of work phases, some of which include Fortum's input. To ensure reliable results, at least two VTT's experts will be independently involved in all the work phases, so that the work results of each expert are reviewed and inspected by another.

An efficient approach to model checking function block based systems is to start by modelling a library of the elementary function blocks [9]. The modular approach makes it very straightforward to construct the models of different I&C functions. Since the **function block library modelling** is the basis of all later models, specific care must be put into writing the code. The model checker is used to verify that the function block models are correct.

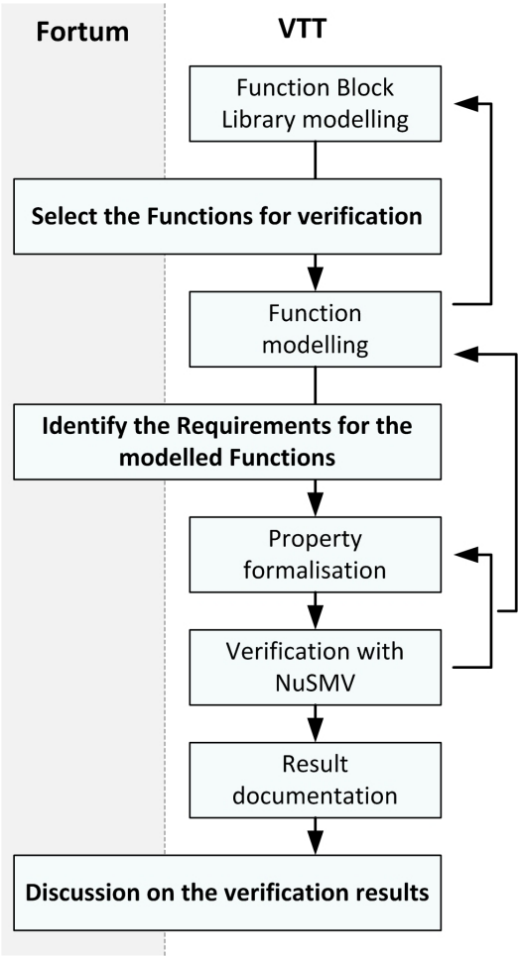

Fig 1. The work process for model checking in LARA

An efficient approach to model checking function block based systems is to start by modelling a library of the elementary function blocks [9]. The modular approach makes it very straightforward to construct the models of different I&C functions. Since the **function block library modelling** is the basis of all later models, specific care must be put into writing the code. The model checker is used to verify that the function block models are correct.

For the system being analysed, the next step is to **select the functions for verification**. Verification of all the functions of each system is not feasible, since analysis of complex control loops is not possible (see section 2.6), and some of the functions are so simple that

manual inspection is sufficient. Due to project constraints, the analysis should also focus on functions directly relevant to safety. The scope of verification is decided by Fortum, with VTT experts providing comments and recommendations.

I&C **function modelling** begins with the careful specification of the model boundary. Specifically, the modeller must consider incorporating "upstream" (i.e. input) processing logic where necessary.

The function modelling task is rather straightforward, with the modeller connecting "invoked objects" of the elementary function block "classes" in a way that reproduces the connections from the original diagrams. VTT tools provide a graphical UI for diagram editing (see chapter 5.4). Still, specific care is needed for, e.g., proper handling of timing, and discretisation of analogue signals (see chapter 2.6).

The next step is to **identify the requirements for the modelled functions**. The documentation that serves as a starting point can be a textual requirement specification document, or other kind of functional description. However, since model checking calls for strict formalisation, there is often a need to further elaborate the requirements. Elaboration is a joint effort between VTT and Fortum. In a renewal project, it is the licensee that has the best knowledge of plant characteristics and functional requirements. As the requirements need to be effectively communicated among Fortum experts – not only I&C but also process engineers – informal textual representation can be used at this point.

At least two VTT experts will work in parallel on **property formalisation**. LTL, CTL and PSL are used. The way that the checked properties are formalised can sometimes be person-dependent, and a good way to ensure that different aspects of requirements are all taken into account is to have different experts working independently.

Verification with NuSMV is also performed concurrently by at least two VTT experts.

If the model checker produces a counterexample that, upon closed inspection of the original design documents, indicates a potential design issue, another VTT expert will first ensure that the result can be duplicated. When confirmed, the issue is reported to Fortum in sufficient detail, so that Fortum experts can reproduce and analyse the scenario.

In any case, the final **result documentation** – a verification report – will identify the method and tools used in the analysis, the target system, the list of documents used in constructing the model and identifying the requirements, and the list of requirements that were verified.

Discussion on the verification results will then take place between VTT and Fortum. Particularly, it is up to Fortum to assess the safety significance of potential findings. It is also possible that careful analysis of plant and I&C system characteristics will show that a reported issue is irrelevant.

It should be noted that the work process is iterative. If a "false" counterexample reveals an error or an omission in the specified models or the requirement formalisation, the modeller may have to revisit earlier work phases. Any modifications will again be reviewed by another expert.

5.4 Tools

The open source model checker NuSMV [6] is used in the analysis.

VTT has developed a dedicated tool to support the model checking of function block diagrams, based on the open source modelling and simulation platform Simantics [9][17]. The tool allows the user to specify the model with a graphical user interface, where function blocks can be added in drag-and-drop fashion and wired together. The resulting diagrams can be encapsulated within composite function block types, and reused on a higher hierarchical level. The use of composite blocks is especially useful in the nuclear context, where similar logic is repeated on parallel subsystems (redundancy principle). Manual coding is only needed when first constructing the library of elementary function block types. When formalising the requirements, references to exact model signals can be copied from the graphical view. The model can then be transformed into the input language used by NuSMV, and verified.

A key advantage of the tool is that the counter-examples output by NuSMV are visualised using an animation that highlights how model signal values change over time. The user is able to freely browse the animation back and forth. Binary signal values are shown by changing the colour and thickness of the block connection wire. Other signal values are shown

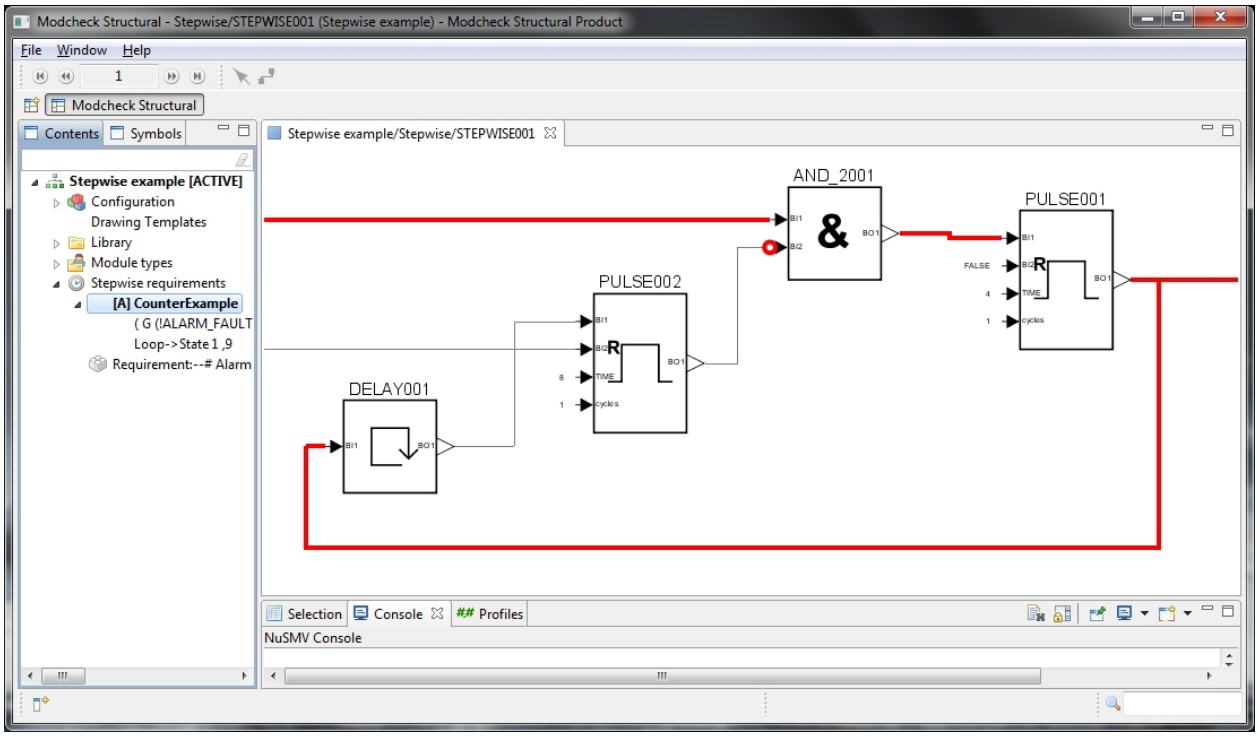

Fig 2. In VTT's tools, function block diagrams are modelled with a graphical UI. The model is transformed into the input language of NuSMV, the model checker is run, and the counterexamples are visualised with an animation.

with monitors attached to the input gates of each block.

Fig. 2 shows one frame (or time step) from a counterexample animation. The example used for the figure is based on the stepwise shutdown logic introduced in [12].

5.5 Results

In Fortum's view, model checking has proved to be a rigorous method for verifying complex I&C functions. The key benefit is exhaustive verification, covering all possible input signal sequences. This means that also events occurring within very small time windows can be analysed, which is challenging when using more traditional verification methods. Furthermore, the mere act of constructing the model (using the precision required by the method) can reveal errors and inconsistencies in design documentation that are easily missed upon manual review.

6 Conclusions

Model checking has been proven a valuable verification tool in different safety critical domains, nuclear included. In the Finnish nuclear industry, we can argue that model checking is already a well-established and integral part of software verification processes, used by both a licensee and the regulator.

Independent verification of software-based systems commissioned by the either the licensee or the vendor is required by several European regulators, among them the Finnish STUK. Accordingly, VTT provides a third-party service for Fortum, using model checking to verify I&C application software in the Loviisa NPP automation renewal project.

The obvious advantage of model checking is exhaustive analysis, which makes it possible to find errors that more conventional verification methods easily miss. Still, proper use of formal methods always calls for careful consideration of the procedures and constraints involved, a point also emphasised by European regulators. In the I&C domain, the inherent limitations mean that model checking is only a supplementary (although valuable) V&V method, and cannot replace, e.g., thorough testing procedures.

Despite the benefits, model checking is far from mainstream use when it comes to I&C. One reason is the manual work needed in constructing the model and updating the model when the system design evolves.

Domain specific tools – such as developed at VTT on the Simantics platform – aim at eventually integrating the method with existing I&C software development tools. Currently, graphical tools make it easy to work with the models, but direct model translation is still an issue for future work. Similarly, tools are needed to ease the formalisation of system properties.

Another reason why model checking is not universally adopted is that generally, I&C engineers, end users, and regulators are simply unaware of the method.

In the nuclear domain, model checking can be utilised by the licensee, or the vendor, as well as the regulatory body. Naturally, the earlier design issues can be identified, the better. In the case of a renewal project, the best knowledge of the plant characteristics and control system requirements lies with the licensee. As the method calls for strict formalisation of functional properties, cooperation between Fortum and VTT on requirement elicitation has been one key to successful application.

Acknowledgement

To a great extent, VTT's research on model checking has been funded by the Finnish Research Programme on Nuclear Power Plant Safety 2011-2014 (SAFIR2014).

References

[1] Guide YVL B.1, Safety Design of a NPP, draft L5, 31.5.2013, STUK 2013.

[2] Licensing of safety critical software for nuclear reactors, Common position of seven European nuclear regulators and authorised technical support organisations, Revision 2013, BEL V, BfS, CSN, ISTec, ONR, SSM & STUK 2013.

[3] E. M. Clarke, Jr., O. Grumberg, D. A. Peled, Model Checking, The MIT Press, 1999.

[4] L. Lamport, "Proving the correctness of multiprocess programs," IEEE Transactions on Software Engineering. Vol. 3, pp. 125-143, 1977.

[5] J. Burch, E. Clarke, K. McMillan, D. Dill, L. Hwang, "Symbolic Model Checking: 1020 States and Beyond," Information and Computation, Vol. 98, pp. 142-170, 1992.

[6] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, "NuSMV 2: An OpenSource Tool for Symbolic Model Checking," International Conference on Computer-Aided Verification (CAV 2002). Copenhagen, Denmark, July 27~31, 2002.

[7] IEC/IEEE 2012. IEC 62531:2012, IEEE Std. 1850, Property Specification Language (PSL), June 2012.

[8] O. Rossi, P. Schnoebelen, "Formal Modeling of Timed Function Blocks for the Automatic Verification of Ladder Diagram Programs," The 4th International Conference on Automation of Mixed Processes (ADPM 2000), Dortmund, Germany, Sep 18~19, 2000.

[9] A. Pakonen, T. Mätäsniemi, J. Lahtinen, T. Karhela, "A Toolset for Model Checking of PLC Software," Proceedings of the 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2013), Cagliari, Italy, Sep. 10~13, 2013.

[10] J. Yoo, S. Cha, E. Jee, "Verification of PLC Programs Written in FBD with VIS," Nuclear Engineering and Technology, Vol. 41, pp. 79-90, 2009.

[11] J. Lahtinen, J. Valkonen, K. Björkman, J. Frits, I. Niemelä, K. Heljanko, "Model checking of safety critical software in the nuclear engineering domain," Reliability Engineering and System Safety. Elsevier. Vol. 105, pp. 104-113, 2012.

[12] K. Björkman, J. Frits, J. Valkonen, K. Heljanko, I. Niemelä, "Model-Based Analysis of a Stepwise Shutdown Logic – MODSAFE 2008 Work Report," VTT Working Papers 115, VTT, 2009.

[13] J. Lahtinen, K. Björkman, J. Valkonen, J. Frits, I. Niemelä, "Analysis of an emergency diesel generator control system by compositional model checking – MODSAFE 2010 Work Report," VTT Working Papers 156, VTT, 2010.

[14] National Nuclear Power Plant Safety Research 2011-2014, SAFIR2014 Framework Plan, The Ministry of the Employment and the Economy, 2010.

[15] E. Jee, S. Jeon, S. Cha, K. Koh, J. Yoo, G. Park, P. Seong, "FBDVerifier: Interactive and Visual Analysis of Counter-Example in Formal Verification of Function Block Diagram," Journal of Research and Practice in Information Technology, Vol. 42, pp. 171-188, 2010.

[16] T. Tommila, A. Pakonen, "Controlled natural language requirements in the design and analysis of safety critical I&C systems," Research report VTT-R-01067-14, VTT, 2014.

[17] T. Karhela, A. Villberg, H. Niemistö, "Open Ontology-Based Integration Platform for Modeling and Simulation in Engineering," International Journal of Modeling, Simulation, and Scientific Computing, Vol. 3, 2012.